Renesas USB MCU

R01AN0401JJ0220
Rev.2.20
Sep 30, 2015

USB Peripheral Human Interface Device Class Driver

要旨

本アプリケーションノートは、USB Peripheral ヒューマンインタフェースクラスドライバ（PHID）について説明します。本ドライバは USB Basic Peripheral Driver(USB-BASIC-FW)と組み合わせることで動作します。以降、本ドライバを PHID と称します。

対象デバイス

RX62N/RX621 グループ
RX630 グループ
RX63T グループ

本アプリケーションノートを他のマイコンへ適用する場合、そのマイコンの仕様にあわせて変更し、十分評価してください。

関連ドキュメント

1. Universal Serial Bus Revision 2.0 specification
2. RX62N/RX621 グループユーザーズマニュアル ハードウェア編 (ドキュメント No.R01UH0033JJ）
3. RX630 グループユーザーズマニュアル ハードウェア編 (ドキュメント No.R01UH0040JJ）
4. RX63T グループユーザーズマニュアル ハードウェア編 (ドキュメント No.R01UH0238JJ)
5. USB Basic Host and Peripheral Driver アプリケーションノート(ドキュメント No. R01AN0512JJ)

— ルネサス エレクトロニクスホームページ
【http://japan.renesas.com/】
— USB デバイスページ
【http://japan.renesas.com/prod/usb/】

目次

1. 概要

PHID は、USB-BASIC-FWと組み合わせることで、USB Peripheral ヒューマンインタフェースデバイスクラスドライバ（以降 PHID と記述）として動作します。PHID は、USB ヒューマンインタフェースデバイスクラス仕様（以降 HID と記述）に準拠し、USB Host との通信を行うことができます。

以下に、本モジュールがサポートしている機能を示します。

- USB Host とのデータ送受信
- HID クラスリクエストに応答
- USB Host からの機能照会に対する応答

1.1 動作確認済環境

PHID の動作確認済環境を以下に示します。

＜評価ボード＞

Renesas Starter Kit+ for RX62N（RSK+RX62N): 型名: R0K5562N0C001BR
RX62N グループ ルネサスマイコン開発スタータキット　ルネサスエレクトロニクス製

Renesas Starter Kit for RX630（RSKRX630): 型名: R0K505630C001BR
RX630 グループ ルネサスマイコン開発スタータキット　ルネサスエレクトロニクス製

Renesas Starter Kit for RX63T（RSKRX63T): 型名: R0K5563THC010BR
RX63T グループ ルネサスマイコン開発スタータキット　ルネサスエレクトロニクス製

＜開発環境＞

a) 統合環境 e^2 studio　ルネサスエレクトロニクス製
b) RX ファミリ用 C/C++コンパイラパッケージ　Ver2.03.00　ルネサスエレクトロニクス製
c) E1 エミュレータまたは E20 エミュレータ　ルネサスエレクトロニクス製

＜その他＞

a) HID ホスト (PC: Windows® 7、Windows® 8、Windows® 8.1、Windows® 10)
b) エミュレータ用ホスト PC (Windows® 7、Windows® 8、Windows® 8.1）
c) USB ケーブル
d) ユーザケーブル（E1 エミュレータまたは E20 エミュレータに同梱）
e) エミュレータ用ケーブル（E1 エミュレータまたは E20 エミュレータに同梱）

用語一覧

本資料で使用される用語と略語は以下のとおりです。

用語		説明
APL	:	Application program
API	:	Application programing Interface
cstd	:	Peripheral & Host USB-BASIC-FW用の関数およびファイルのプレフィックス
HID	:	Human Interface Device class
USB Host	:	HID class USB Host
HM	:	Hardware Manual
PCD	:	Peripheral control driver of USB-BASIC-FW
PDCD	:	Peripheral device class driver (device driver and USB class driver)
PHID	:	Peripheral Human Interface Devices
PP	:	プリプロセス定義
pstd	:	Peripheral USB-BASIC-FW用の関数およびファイルのプレフィックス
RSK	:	Renesas Starter Kits
USB	:	Universal Serial Bus
USB-BASIC-FW	:	USB Basic Peripheral Driver
スケジューラ	:	タスク動作を簡易的にスケジューリングするもの
スケジューラマクロ	:	スケジューラ機能を呼び出すために使用されるもの
タスク	:	処理の単位
データ転送	:	Control転送、Bulk転送、Interrupt転送の総称

2. ソフトウェア構成

Figure 2-1に PHID のモジュール構成、Table 2.1にモジュール機能概要を示します。

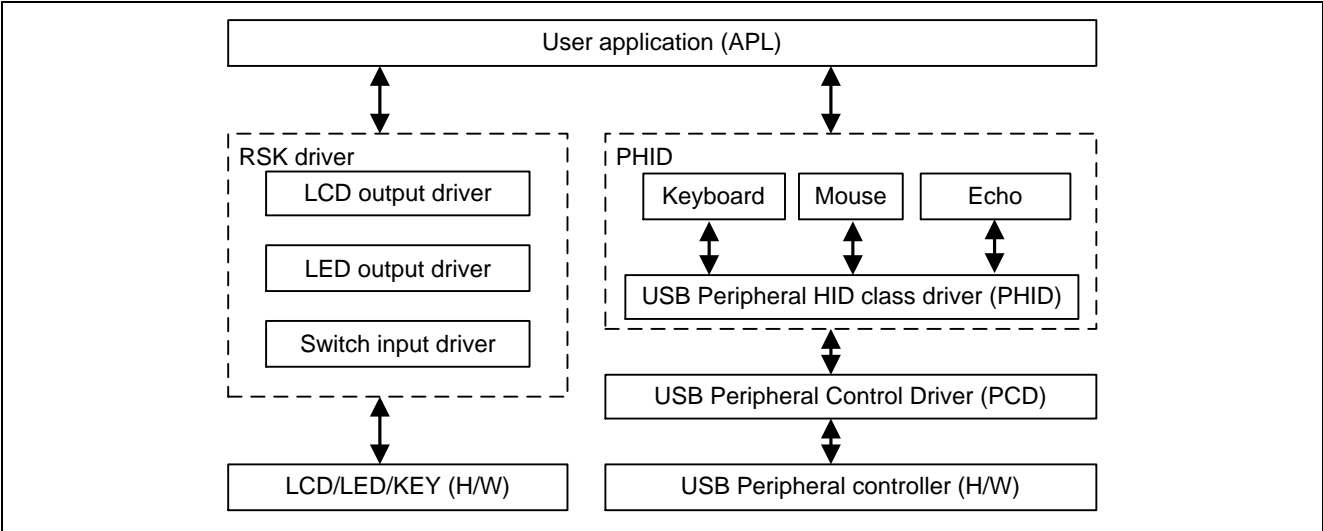



Figure 2-1 モジュール構成図

Table 2.1 各モジュール機能概要

モジュール名	機能概要
PHID	USB Host からの要求を解析します。 APL からの要求を PCD 経由で USB Host に通知します。
USB-Basic-FW	USB ペリフェラルコントロールドライバです。 （ハードウェア制御とデバイスステート管理を行います。）

2.1 API 情報

本ドライバの API はルネサスの API の命名基準に従っています。

2.1.1 ハードウェアの要求

ご使用になる MCU が以下の機能をサポートしている必要があります。

- USB

2.1.2 サポートされているツールチェイン

このドライバは下記ツールチェインで動作確認を行っています。

- Renesas RX Toolchain　v.2.03.00

2.1.3 ヘッダファイル

すべての API 呼び出しとそれをサポートするインタフェース定義は r_usb_phid_if.h に記載しています。

2.1.4 整数型

このプロジェクトは ANSI C99 を使用しています。これらの型は stdint.h で定義されています。

2.1.5 コンパイル時の設定

本モジュールを使用する場合、USB-BASIC-FWをペリフェラルとして設定する必要があります。USB-BASIC-FWの設定は、USB Basic Host and Peripheral Driver アプリケーションノート(Document No.R01AN0512JJ)を参照してください。

本モジュールのコンフィグレーションオプションの設定は、r_usb_phid_config.h で行います。

オプション名および設定値に関する説明を、下表に示します。

Configuration options in r_usb_phid_config.h	
USB_PHID_MODE	HID デバイスのモードを選択します。 Keyboard mode：USB_PHID_KEYBOARD_MODE Mouse mode：USB_PHID_MOUSE_MODE Echo mode：USB_PHID_ECHO_MODE
USB_PHID_USE_PIPE_IN	USB IN データ転送で使用するパイプ番号を指定してください。 (USB_PIPE6 から USB_PIPE9 のいずれかを指定してください。)
USB_PHID_USE_PIPE_OUT	USB OUT データ転送で使用するパイプ番号を指定してください。 (USB_PIPE6 から USB_PIPE9 のいずれかを指定してください。)
USB_REPORT_DISCRIPTOR	Report Descriptor を記述した領域の先頭アドレスを指定してください。 (上記先頭アドレスが変数の場合、当該変数の extern 宣言の記述もお願いします。)
USB_PHID_FUNC	SetReport クラスリクエスト受信時の振る舞いを記述したアプリケーション関数を定義してください。当該関数には"uint8_t *"型の引数を一つ定義してください。 (上記アプリケーション関数のプロトタイプ宣言の記述もお願いします。)

[Note]

1. "Echo mode"では、MaxPacketSize が 64 バイトの Interrupt IN and Interrupt OUT を使ったデータ転送を行います。
2. USB_PHID_USE_PIPE_IN と USB_PHID_USE_PIPE_OUT には､異なるパイプ番号を指定してください。

3. "Keyboard mode"または"Echo mode"を選択したとき、USB_PHID_USE_PIPE_OUT に対してパイプ番号を指定してください。

2.1.6 引数

API 関数の引数である構造体を示します。詳細は USB Basic Host and Peripheral Driver アプリケーションノート(Document No.R01AN0512JJ)を参照してください。

```
struct USB_UTR_t
{
    USB_MH_t            msghead;        /* OSが使用するメッセージヘッダ/使用不可　*/
    uint16_t            msginfo;        /* USB-BASIC-F/Wが使用するメッセージ情報　*/
    uint16_t            keyword;        /* サブ情報(ポート番号、パイプ番号) */
    union {
        USB_REGADR_t    ipp;            /* USB IPアドレス(USBbモジュール用)*/
        USB_REGADR1_t   ipp1;           /* USB IPアドレス(USBA/USBAaモジュール用) */
    };
    uint16_t            ip;             /* USBIP番号*/
    uint16_t            result;         /* USB通信結果 */
    USB_CB_t            complete;       /* コールバック関数 */
    void                *tranadr;       /* USB通信バッファアドレス */
    uint32_t            tranlen;        /* USB通信データ長 */
    uint16_t            *setup;         /* セットアップパケットデータ */
    uint16_t            status;         /* USB通信ステータス */
    uint16_t            pipectr;        /* PIPECTRレジスタ */
    uint8_t             errcnt;         /*転送中のエラー発生回数 */
    uint8_t             segment;        /* セグメント情報（データ通信の継続又は終了） */
    void                *usr_data;      /* 各種情報を格納 */
}
```

3.　デバイスクラスドライバの登録

作成したデバイスクラスドライバは、USB ドライバに登録することでデバイスクラスドライバとして機能します。

詳細は、USB Basic Host and Peripheral Driver アプリケーションノート(Document No.R01AN2025JJ)の「ペリフェラルクラスドライバの登録方法」を参照してください。

4.　システム資源

Table 4.1、Table 4.2に、PHID が使用しているタスク ID を示します。

Table 4.1　タスク情報

関数名	タスク ID	優先度	概要
usb_phid_Task	USB_PHID_TSK	USB_PRI_1	PHID タスク

Table 4.2　メールボックス情報

メールボックス名	使用タスク ID	待ちタスクキュー	概要
USB_PHID_MBX	USB_PHID_TSK	FIFO 順	PHID 用メールボックス

5. ヒューマンインタフェースデバイスクラス（HID）

5.1 基本機能

本ソフトウェアは Human Interface Device Class(HID)仕様に準拠します。
PHID の主な機能は、以下のとおりです。

1. USB Host からの機能照会に対する応答
2. USB Host からのクラスリクエストに対する応答
3. USB Host とのレポート送信

5.2 HID デバイスクラス概要

5.2.1 クラスリクエスト（ホスト→デバイスへの通知）

PHID で対応しているクラスリクエストをTable 5.1に示します。

Table 5.1　HID クラスリクエスト

リクエスト	コード	説明	対応
Get_Report	0x01	USB Host へレポートを送信する。	×
Set_Report	0x09	USB Host からのレポートを受信する。	×
Get_Idle	0x02	USB Host へ Duration 時間を送信する。	×
Set_Idle	0x0A	USB Host からの Duration 時間設定を受信する。	×
Get_Protocol	0x03	USB Host へプロトコルを送信する。	×
Set_Protocol	0x0B	USB Host からのプロトコルを受信する。	×
Get_Desctiptor Descriptor Type：Class Class Descriptor Type：Report	0x06 (Standard)	レポートディスクリプタを送信する。	○
Get_Desctiptor Descriptor Type：Class Class Descriptor Type：HID	0x06 (Standard)	HID ディスクリプタを送信する。	○

※詳細は“USB Device Class Definitions for Human Interface Device, Revision1.1”の 7 章を参照ください。

6. USB ペリフェラルコミュニケーションデバイスクラスドライバ（PHID）

6.1 基本機能

PHID の基本機能を以下に示します。

1. USB Host とのデータ送受信
2. HID クラスリクエストに対する応答

6.2 PHID API 一覧

Table 6.1に PHID API 一覧を示します。

Table 6.1 PHID API 一覧

関数名	機能概要
R_usb_phid_send_data	USB 送信処理
R_usb_phid_receive_data	USB 受信処理
R_usb_phid_DeviceInformation	デバイス状態の情報取得
R_usb_phid_driver_start	PHID クラスドライバタスクの開始
R_usb_phid_TransferEnd	USB 転送強制終了
R_usb_phid_task	PHID タスク処理
R_usb_phid_control_transfer	PHID 用コントロール転送処理

6.2.1 R_usb_phid_send_data

USB 送信処理

形式

void	R_usb_phid_send_data(	USB_UTR _t	*ptr,
		uint8_t	*buf,
		uint32_t	size,
		USB_CB_t	complete)

引数

ptr	USB 通信用構造体領域へのポインタ
buf	転送データを格納した領域へのポインタ
size	転送サイズ
complete	処理完了通知コールバック関数

戻り値

—	—

解説

データ送信要求を行います。

転送データアドレス"buf"で指定されたアドレスから、転送サイズ“size”分のデータを USB 送信します。

送信完了後、コールバック関数 complete が呼出されます。

補足

1. ユーザアプリケーションから本 API を呼び出してください。
2. USB 通信用構造体 USB_UTR_t の以下のメンバ設定が必要です。

USB_REGADR_t	ipp	: USB IP のアドレス
uint16_t	ip	: USB IP 番号

3. 第 2 引数には自動変数(スタック)領域以外の領域を指定してください。
4. USB 通信用構造体(USB_UTR_t 構造体）については2.1.6章を参照してください。

使用例

```
void usb_apl( void )
{
                :
  R_usb_phid_send_data(&hid_utr, buf, size, (USB_CB_t)usb_complete);
                :
}

/* USB 送信完了通知用コールバック関数 */
void  usb_complete( USB_UTR_t *mess, uint16_t data1, uint16_t data2 )
{
  /* USB 送信完了時の処理を記述してください。 */
}
```

6.2.2 R_usb_phid_receive_data

USB 受信処理

形式

```
void        R_usb_phid_receive_data( USB_UTR _t   *ptr,
                                     uint8_t      *buf,
                                     uint32_t     size,
                                     USB_CB_t     complete )
```

引数

ptr	USB 通信用構造体領域へのポインタ
buf	転送データアドレス
size	転送サイズ
complete	処理完了通知コールバック関数

戻り値

－　　－

解説

データ受信要求を行います。
転送データアドレス“buf”で指定されたアドレスから、転送サイズ“size”分のデータを USB 送信します。
送信完了後、コールバック関数 complete が呼出されます。

補足

1. ユーザアプリケーションから本 API を呼び出してください。
2. USB 通信用構造体 USB_UTR_t の以下のメンバ設定が必要です。
   USB_REGADR_t　ipp　：USB IP のアドレス
   uint16_t　ip　：USB IP 番号
3. 第 2 引数には自動変数(スタック)領域以外の領域を指定してください。
4. 受信したデータが MaxPacketSize の n 倍、かつ引数 size に指定したサイズに満たない場合は、データ転送の途中であると判断しコールバック関数 complete は発生しません。
5. USB 送信処理結果はコールバック関数の引数“USB_UTR_t *”で得られます。USB_UTR_t 構造体のメンバ tranlen には、残りの転送サイズがセットされます。
6. USB 通信用構造体（USB_UTR_t 構造体）については2.1.6章を参照してください。

使用例

```
void usb_apl( void )
{
            :
  R_usb_phid_receive_data(&hid_utr,buf,size,(USB_CB_t)&usb_complete);
            :
}

/* USB 受信完了通知用コールバック関数 */
void  usb_complete( USB_UTR_t *mess, uint16_t data1, uint16_t data2 )
{
  /* USB 受信完了時の処理を記述してください。 */
}
```

6.2.3 R_usb_phid_DeviceInformation

デバイス状態の情報取得

形式

void　　R_usb_phid_DeviceInformation(USB_UTR_t *ptr, uint16_t *deviceinfo)

引数

*ptr　　USB 通信用構造体領域へのポインタ
*deviceinfo　　デバイスステート格納用バッファへのポインタ

戻り値

—　　—

解説

与えられた USB ペリフェラルのデバイス情報を取得します。デバイス情報は引数“deviceinfo”で指定されたアドレスへ格納されます。

[0] ：USB デバイスステート *1

b15-b8	未使用	
b7	VBSTS	VBUS 入力ステータスビット 0：USB0_VBUS 端子が Low 1：USB0_VBUS 端子が High
b6-b4	DVSQ[2:0]	デバイスステートビット 000：パワードステート 001：デフォルトステート 010：アドレスステート 011：コンフィグレーションステート 1xx：サスペンドステート　x：Don’t care

*1：詳細はハードウェアマニュアルの「割り込みステータスレジスタ 0（INTSTS0）」を参照してください。

[1] ：USB 転送速度
0x0000：未接続
0x00C0：Hi-Speed 接続（非サポート）
0x0080：Full-Speed 接続
0x0040：Low-Speed 接続

[2] ：使用しているコンフィグレーション番号
[3] ：使用しているインタフェース番号
[4] ：リモートウェイクアップフラグ（0：ウェイクアップ制御禁止、1：ウェイクアップ制御許可）

補足

1. ユーザアプリケーションから本 API を呼び出してください。
2. USB 通信用構造体 USB_UTR_t の以下のメンバ設定が必要です。

USB_REGADR_t	ipp	: USB IP のアドレス
uint16_t	ip	: USB IP 番号

使用例

```
USB_UTR_t       hid_utr;
void usb_apl( void )
{
   uint16_t res[5];
   :
  /* USB デバイス情報を取得 */
  R_usb_pstd_DeviceInformation(&hid_utr, (uint16_t *)res);
   :
}
```

6.2.4 R_usb_phid_driver_start

PHID クラスドライバタスクの開始

形式

void R_usb_phid_driver_start(void)

引数

— —

戻り値

— —

解説

PHID ドライバを開始します。

補足

1. 初期設定時にユーザアプリケーションで呼び出してください。

使用例

```
void usb_pstd_task_start( void )
{
 usb_phid_driver_registration(); /*  ペリフェラルアプリ Registration*/
 usb_papl_task_start();           /* アプリケーションタスク設定 */
 R_usb_phid_driver_start();       /* クラスドライバタスク設定 */
 R_usb_pstd_usbdriver_start();    /* USB ドライバ設定 */
}
```

6.2.5 R_usb_phid_TransferEnd

USB 転送強制終了

形式

void R_usb_phid_TransferEnd (USB_UTR_t *ptr)

引数

***ptr** USB 通信用構造体領域へのポインタ

戻り値

－ －

解説

パイプ経由でのデータ転送を強制終了します。
本 API をコールすると、PCD に対してデータ転送強制終了要求を行います。PCD は要求を受信すると、データ送信強制終了要求動作を行います。
データ転送が強制的に終了された場合でも、データ転送時に、R_usb_phid_send_data でセットされたコールバック関数が呼び出されます。強制終了の情報として、送信データ長、残りの受信データ長、ステータス、送信エラーのコード番号が、コールバック関数の引数（mess）へ設定されます。

補足

1. ユーザアプリケーションから本 API を呼び出してください。
2. USB 通信用構造体 USB_UTR_t の以下のメンバ設定が必要です。

USB_REGADR_t	ipp	: USB IP のアドレス
uint16_t	ip	: USB IP 番号

3. 第 1 引数には自動変数(スタック)領域以外の領域を指定してください。

使用例

```
USB_UTR_t    hid_utr;
void usb_apl( void )
{
     :
  /* 転送終了リクエスト */
  R_usb_phid_TransferEnd(&hid_utr);
    :
}
```

6.2.6 R_usb_phid_control_transfer

PHID 用コントロール転送処理

形式

void　R_usb_phid_control_transfer(USB_UTR_t *ptr, USB_REQUEST_t *request, uint16_t ctsq)

引数

*ptr	USB 通信用構造体領域へのポインタ
*request	クラスリクエストメッセージのポインタ
ctsq	コントロール転送ステージ情報

USB_CS_IDST	(0x00)	: Idle or setup stage
USB_CS_RDDS	(0x01)	: Control read data stage
USB_CS_RDSS	(0x02)	: Control read status stage
USB_CS_WRDS	(0x03)	: Control write data stage
USB_CS_WRSS	(0x04)	: Control write status stage
USB_CS_WRND	(0x05)	: Control write no data status stage
USB_CS_SQER	(0x06)	: Sequence error

戻り値

—　—

解説

リクエストタイプが HID クラスリクエストの場合、コントロール転送ステージに対応した処理を呼び出します。

本 API をデバイスクラスドライバ・レジストレーションでコントロール転送時に呼び出すコールバック関数として登録してください。

補足

—

使用例

```
void usb_apl( void )
{
  USB_PCDREG_t driver;
    :
  /* コントロール転送 */
  driver.ctrltrans = (USB_CB_TRN_t)&R_usb_phid_control_transfer;
  R_usb_pstd_DriverRegistration(&driver);
    :
}
```

6.2.7 R_usb_phid_task

PHID タスク

形式

void R_usb_phid_task(void)

引数

— —

戻り値

— —

解説

PHID タスクはアプリケーションから要求された処理を行い、アプリケーションに処理結果を通知します。

補足

1. ユーザプログラムから本 API を呼び出してください

使用例

```
void usb_apl(void)
{
  while( 1 )
  {
    if( USB_FLGSET == R_usb_cstd_Scheduler())
    {
       /* PCD Task */
       R_usb_pstd_PcdTask();

       /* Peripheral HID Task */
       R_usb_phid_task();

       /* Peripheral HID Application Task */
       usb_phid_main_task();
    }
  }
}
```

7. サンプルアプリケーション

7.1 アプリケーション仕様

PHID のサンプルアプリケーション(以降、APL)の主な機能を以下に示します。

(1). Keyboard mode: キーボード機能

RSK を USB Host に接続すると、USB Host は RSK をキーボードとして認識します。RSK はキーボードとして動作し、Interrupt IN 転送によりキーボードデータを USB Host に送信します。

(2). Mouse mode: マウス機能

RSK を USB Host に接続すると、USB Host は RSK をマウスとして認識します。RSK はマウスとして動作し、Interrupt IN 転送によりマウスデータを USB Host に送信します。

(3). Echo mode: USB ループバック機能(Interrupt IN/OUT データ転送)

RSK を USB Host に接続し、USB Host との Interrupt IN/OUT のデータ転送を行います。
この機能では、USB Host から受信したデータをそのまま USB Host へ送信する処理を行います。

(4). 消費電力低減機能

USB の状態に応じて MCU を消費電力低減モードに遷移させる機能です。この機能を有効にするには、“r_usb_config.h”ファイル内のマクロ定義“USB_CPU_LPW_PP”を“USB_LPWR_USE_PP”に設定します。

a) USB サスペンド状態時に MCU をスリープモードに遷移させます。
b) USB デタッチ（切断）状態時に、MCU をソフトウエアスタンバイモードに遷移させます。

[Note]

1. Keyboard mode/Mouse mode/Echo mode の選択は、r_usb_phid_config.h 内で行ってください。
2. Echo mode の場合は、USB ループバック機能をサポートしている USB Host と通信を行います。Keyboard mode および Mouse mode の場合は、Windows 7/Windows 8/Windows 8.1/Windows 10 等の OS をサポートしている PC(USB Host)との USB 通信が可能です。

7.2 アプリケーション処理概要

APL は初期化処理、メインループの 2 つの部分から構成されます。以下にそれぞれの処理概要を示します。

7.2.1 初期設定

初期設定では、MCU の端子設定、USB ドライバの設定、USB コントローラの初期設定を行います。

7.2.2 メインループ (Mouse mode)

Mouse mode のメインループでは、以下の処理を行います。

a) USB Host との Enumeration が完了すると、USB_PCDREG_t 構造体のメンバ devconfig に設定した関数 hid_open を USB ドライバがコールします。関数 hid_open では、イベント値に EVENT_CONFIGURED をセットします。

b) EVENT_USB_WRITE_START 処理では、R_usb_phid_send_data 関数をコールし、USB ドライバに対しマウスデータの送信要求を行います。マウスデータの送信が完了すると R_usb_phid_send_data 関数の第 4 引数に指定したコールバック関数 hid_write_trans_cb がコールされます。このコールバック関数では、イベント値に EVENT_USB_WRITE_COMPLETE をセットします。

c) EVENT_USB_WRITE_COMPLETE 処理では、送信状態変数を非送信状態にします。

d) EVENT_SUSPEND 処理では、関数 hid_low_power_control をコールし、HID デバイス(RSK)を低消費電力モード(スリープモード)に移行します。なお、サスペンド状態にある HID デバイス(RSK)は、USB Host が送信するレジューム信号を検出することによって、サスペンド状態から復帰します。

e) EVENT_NONE 処理ではキー入力情報を取得し、SW1 が押下され、

   i. HID デバイスがサスペンド状態であれば R_usb_phid_ChangeDeviceState 関数によって RemoteWakeUp 信号を USB Host に送信します。

   ii. HID デバイスが Configured 状態であれば、キーデータ(マウスデータ)設定処理を行い、設定したデータを USB Host へ送信するためイベント値に EVENT_USB_WRITE_START をセットします。送信状態変数を送信状態にします

f) 上記b)からe)までの処理が繰り返し行われている間に、USB Host が HID デバイス(RSK)に対しサスペンド信号を送信すると、USB ドライバは、USB_PCDREG_t 構造体のメンバ devsuspend に設定した関数 hid_suspend をコールします。関数 hid_suspend では、イベント値に EVENT_SUSPEND をセットします。

g) EVENT_SUSPEND 処理では、関数 hid_low_power_control をコールし、HID デバイス(RSK)を低消費電力モード(スリープモード)に移行します。なお、サスペンド状態にある HID デバイス(RSK)は、USB Host が送信するレジューム信号を検出することによって、サスペンド状態から復帰します。

h) 上記b)からe)までの処理が繰り返し行われている間に、HID デバイス(RSK)が USB Host からデタッチされると、USB ドライバは、USB_PCDREG_t 構造体のメンバ devdetach に設定した関数 hid_close をコールします。関数 hid_close では、イベント値に EVENT_DETACH をセットします。

i) EVENT_DETACH 処理では、関数 R_usb_phid_TransferEnd をコールし、データ送信要求を停止後、低消費電力モード(ソフトウェアスタンバイモード)に移行します。再度、USB Host に HID デバイス(RSK)をアタッチすると USB Host との Enumeration が開始され、上記a)から処理が再開されます。以下に、APL の処理概要を示します。

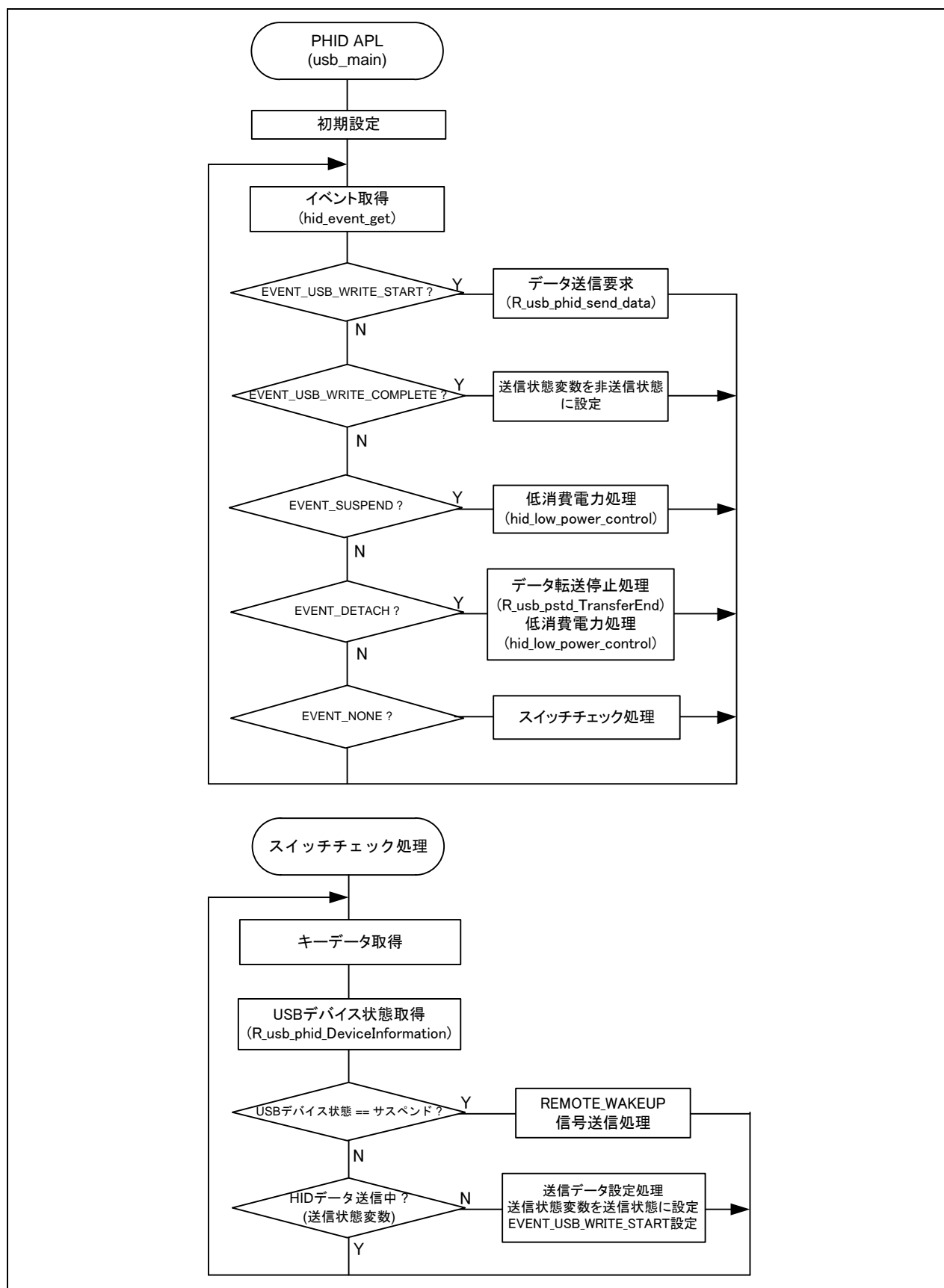



Figure 7-1　メインループ処理 (Mouse mode)

7.2.3 メインループ (Keyboar mode)

Keyboard mode のメインループでは、以下の処理を行います。

a) USB Host との Enumeration が完了すると、USB_PCDREG_t 構造体のメンバ devconfig に設定した関数 hid_open を USB ドライバがコールします。関数 hid_open では、イベント値に EVENT_CONFIGURED をセットします。

b) EVENT_CONFIGURED 処理では、R_usb_phid_receive_data 関数をコールし、USB ドライバに対し USB Host から送信される OUT データの受信要求を行います。OUT データの受信が完了すると R_usb_phid_receive_data 関数の第 4 引数に指定したコールバック関数 hid_read_trans_cb がコールされます。このコールバック関数では、イベント値に EVENT_USB_READ_COMPLETE をセットします。

c) EVENT_USB_READ_COMPLETE 処理では、受信した OUT データをもとにした LED 点灯処理等を行います。また、イベント値に EVENT_USB_READ_START をセットします。

d) EVENT_USB_WRITE_START 処理では、R_usb_phid_send_data 関数をコールし、USB ドライバに対し USB Host へキー入力データ等の送信要求を行います。キー入力データ等の送信が完了すると R_usb_phid_send_data 関数の第 4 引数に指定したコールバック関数 hid_write_trans_cb がコールされます。このコールバック関数では、イベント値に EVENT_USB_WRITE_COMPLETE をセットします。

e) EVENT_USB_WRITE_COMPLETE 処理では、

   i. キー入力データ送信完了時の場合、キー押下リリース(キーが離された)を USB Host に通知するため、8 バイト分の 0 データを USB Host へ送信する必要があります。このための 0 データ設定処理を行い、送信状態変数には 0 データ送信状態を設定します。また、イベント値には EVENT_USB_WRITE_START をセットします。

   ii. 0 データ送信完了時の場合、送信状態変数を非送信状態に設定します。

f) EVENT_NONE 処理ではキー入力が行われたかどうかをチェックします。SW1 が押下→リリースされ、HID デバイスがサスペンド状態であれば R_usb_phid_ChangeDeviceState 関数によって RemoteWakeUp 信号を USB Host に送信します。SW1 以外が押下された場合は、キーデータ設定処理を行い、送信状態変数をキーデータ送信状態に設定します。また、設定したキーデータを USB Host へ送信するためイベント値に EVENT_USB_WRITE_START をセットします。

g) 上記b)からf)までの処理が繰り返し行われている間に、USB Host が HID デバイス(RSK)に対しサスペンド信号を送信すると、USB ドライバは、USB_PCDREG_t 構造体のメンバ devsuspend に設定した関数 hid_suspend をコールします。関数 hid_suspend では、イベント値に EVENT_SUSPEND をセットします。

h) EVENT_SUSPEND 処理では、関数 hid_low_power_control をコールし、HID デバイス(RSK)を低消費電力モード(スリープモード)に移行します。なお、サスペンド状態にある HID デバイス(RSK)は、USB Host が送信するレジューム信号を検出することによって、サスペンド状態から復帰します。

i) 上記b)からf)までの処理が繰り返し行われている間に、HID デバイス(RSK)が USB Host からデタッチされると、USB ドライバは、USB_PCDREG_t 構造体のメンバ devdetach に設定した関数 hid_close をコールします。関数 hid_close では、イベント値に EVENT_DETACH をセットします。

j) EVENT_DETACH 処理では、関数 R_usb_phid_TransferEnd をコールし、データ送受信要求を停止後、低消費電力モード(ソフトウェアスタンバイモード)に移行します。再度、USB Host に HID デバイス(RSK)をアタッチすると USB Host との Enumeration が開始され、上記a)から処理が再開されます。以下に、APL の処理概要を示します。

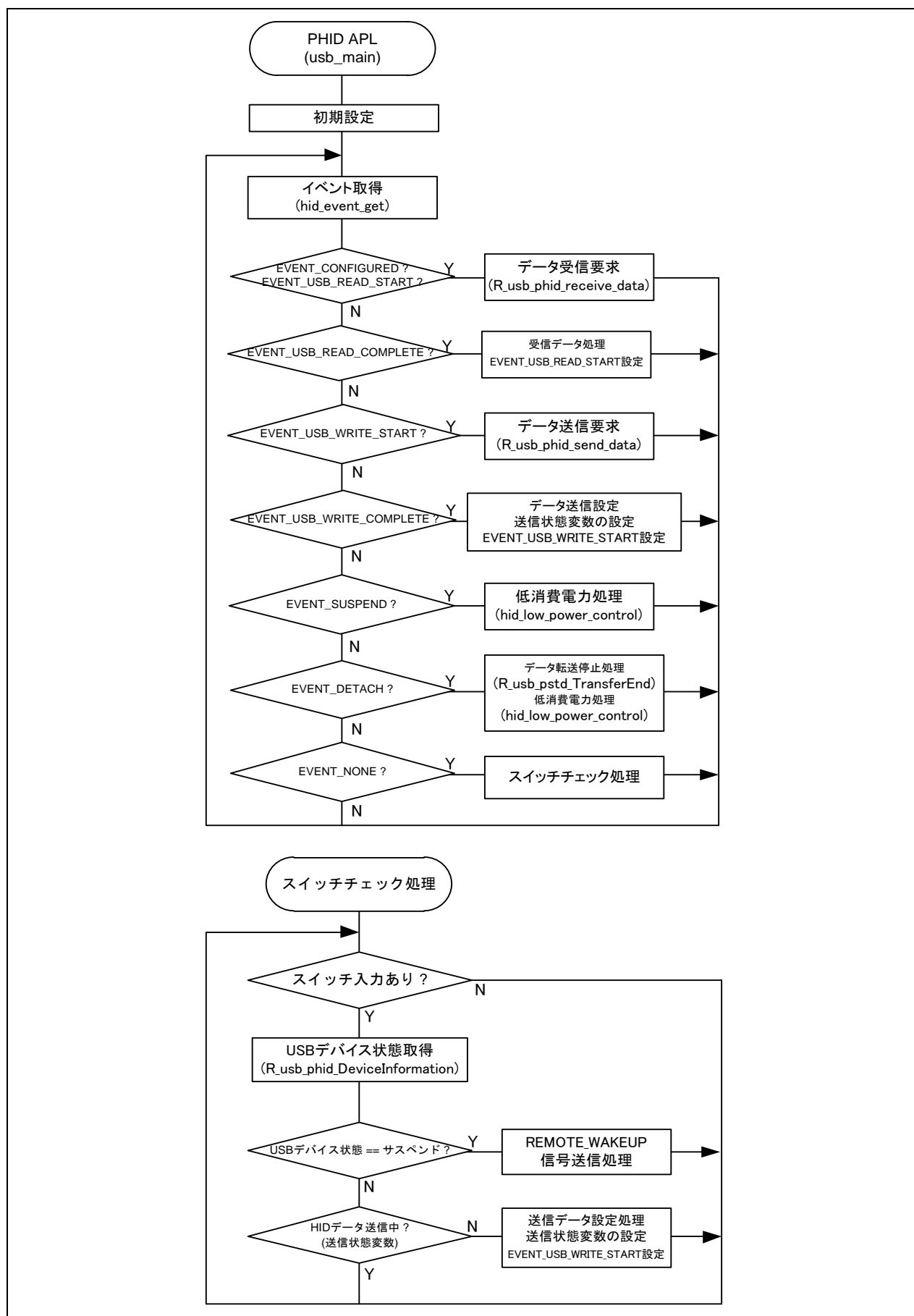



Figure 7-2　メインループ処理 (Keyboard mode)

7.2.4 メインループ (Echo mode)

Echo mode のメインループでは、以下の処理を行います。

a) USB Host との Enumeration が完了すると、USB_PCDREG_t 構造体のメンバ devconfig に設定した関数 hid_open を USB ドライバがコールします。関数 hid_open では、イベント値に EVENT_CONFIGURED をセットします。

b) EVENT_CONFIGURED 処理では、USB Host からのデータを受信するため R_usb_phid_receive_data 関数をコールします。データ受信が完了すると R_usb_phid_receive_data 関数の第 4 引数に指定したコールバック関数 hid_read_trans_cb がコールされます。このコールバック関数では、イベント値に EVENT_USB_READ_COMPLETE をセットします。

c) EVENT_USB_READ_COMPLETE 処理では、上記b)で受信したデータを USB Host に送信するため R_usb_phid_send_data 関数をコールします。データ送信が完了すると R_usb_phid_send_data 関数の第 4 引数に指定したコールバック関数 hid_write_trans_cb がコールされます。このコールバック関数では、イベント値に EVENT_USB_WRITE_COMPLETE をセットします。

d) EVENT_USB_WRITE_COMPLETE 処理では、イベント値に EVENT_USB_READ_START をセットします。

e) 上記b)からd)までの処理が繰り返し行われている間に USB Host が HID デバイス(RSK)に対しサスペンド信号を送信すると、USB ドライバは、USB_PCDREG_t 構造体のメンバ devsuspend に設定した関数 hid_suspend をコールします。関数 hid_suspend では、イベント値に EVENT_SUSPEND をセットします。

f) EVENT_SUSPEND 処理では、関数 hid_low_power_control をコールし、HID デバイス(RSK)を低消費電力モード(スリープモード)に移行します。なお、サスペンド状態にある HID デバイス(RSK)は、USB Host が送信するレジューム信号を検出することによって、サスペンド状態から復帰します。

g) 上記b)からd)までの処理が繰り返し行われている間に、HID デバイス(RSK)が USB Host からデタッチされると、USB ドライバは、USB_PCDREG_t 構造体のメンバ devdetach に設定した関数 hid_close をコールします。関数 hid_close では、イベント値に EVENT_DETACH をセットします。

h) EVENT_DETACH 処理では、関数 R_usb_phid_TransferEnd をコールし、データ送受信要求を停止後、低消費電力モード(ソフトウェアスタンバイモード)に移行します。USB Host に HID デバイス(RSK)をアタッチすると USB Host との Enumeration が開始され、上記a)から処理が再開されます。以下に、APL の処理概要を示します。

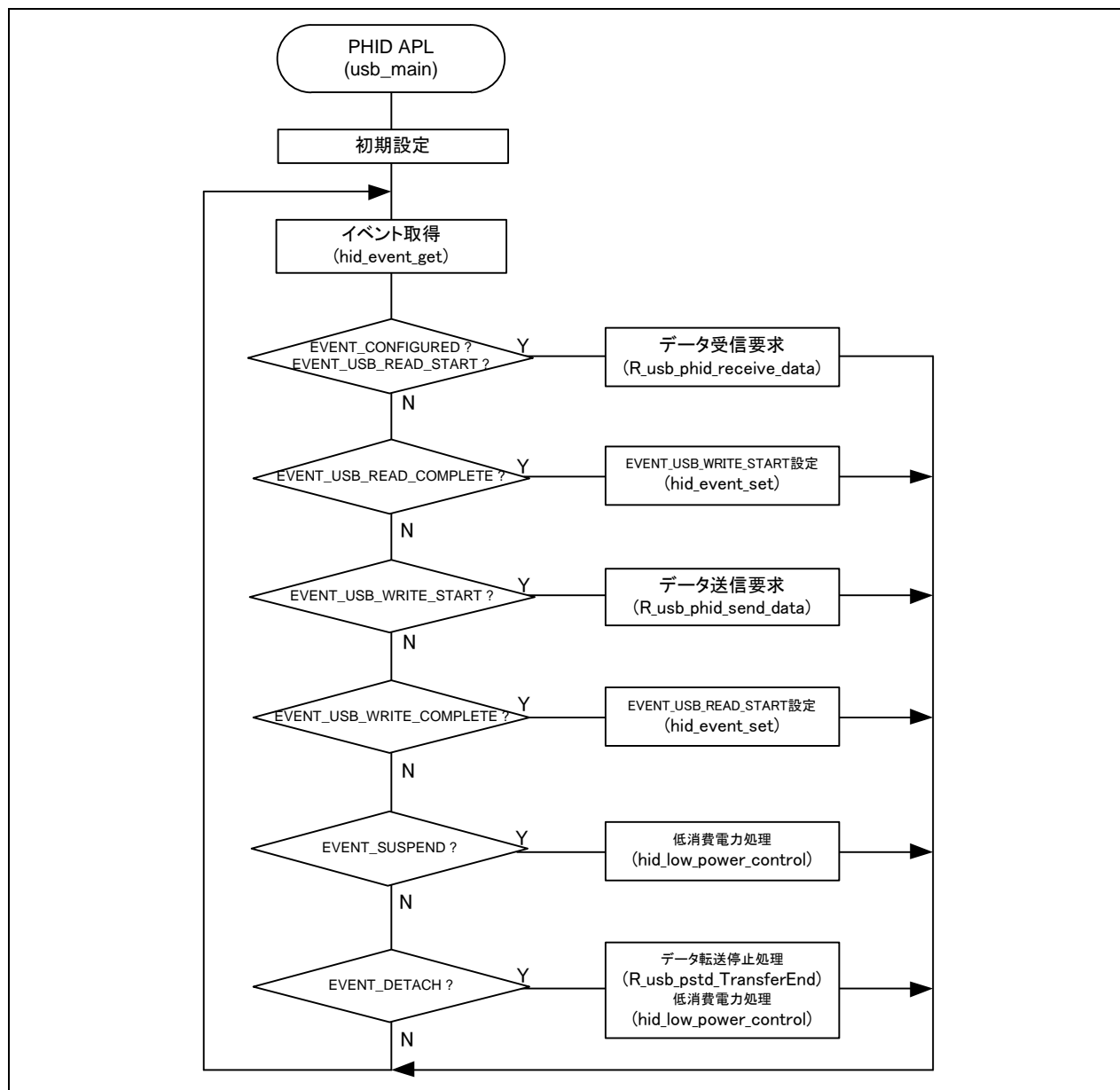



Figure 7-3　メインループ処理 (Echo mode)

Table 7-1 イベント一覧

イベント	概要
EVENT_CONFIGURD	USB デバイス接続完了
EVENT_USB_READ_START	データ受信要求
EVENT_USB_READ_COMPLETE	データ受信完了
EVENT_USB_WRITE_START	データ送信要求
EVENT_USB_WRITE_COMPLETE	データ送信完了
EVENT_SUSPEND	サスペンド
EVENT_DETACH	USB デタッチ
EVENT_NONE	イベントなし

[Note]

1. hid_event_get 関数は取り出すイベントが存在しない場合、“EVENT_NONE”を返します。
2. 以下のイベントはコールバック関数によって設定されます。各イベントを設定するコールバック関数を以下に示します。

Table 7-2 イベントに関連するコールバック関数

イベント	コールバック関数
EVENT_CONFIGURD	hid_open
EVENT_USB_READ_COMPLETE	hid_read_trans_cb
EVENT_USB_WRITE_COMPLETE	hid_write_trans_cb
EVENT_SUSPEND	hid_suspend
EVENT_DETACH	hid_close

7.2.5 イベントの管理

イベントは、以下の構造体のメンバ(*event_cnt, event[]*)によって管理されています。この構造体は、APL が用意している構造体です。

```
typedef struct DEV_INFO                     /* Structure for HID device control */
{
        uint16_t    event_cnt;              /* Event count */
        uint16_t    event[EVENT_MAX];       /* Event. */
} DEV_INFO_t;
```

7.2.6 MCU 消費電力低減処理

MCU 消費電力低減処理は、Table 7-3の条件が成立すると消費電力低減モードに移行する処理を行います。

Table 7-3 消費電力低減機能状態遷移条件

遷移条件		遷移状態
VBUS	USB ステート	
OFF	―	ソフトウエアスタンバイモード
ON	Suspend Configured	スリープモード
ON	Suspend Configured 以外	通常モード（プログラム実行状態）

a) デタッチ（VBUS OFF）状態の場合、APL は MCU をソフトウエアスタンバイモードに遷移します。なお、ソフトウエアスタンバイモードからの復帰は、HID デバイス(RSK)を USB Host にアタッチすることにより通常モードに移行します。

b) HID デバイス(RSK)を USB Host に接続した状態で、USB Host から送信されるサスペンド信号を受信すると APL は、MCU をスリープモードに遷移します。なお、スリープモードからの復帰は、USB Host から送信されるレジューム信号の受信により行われます。

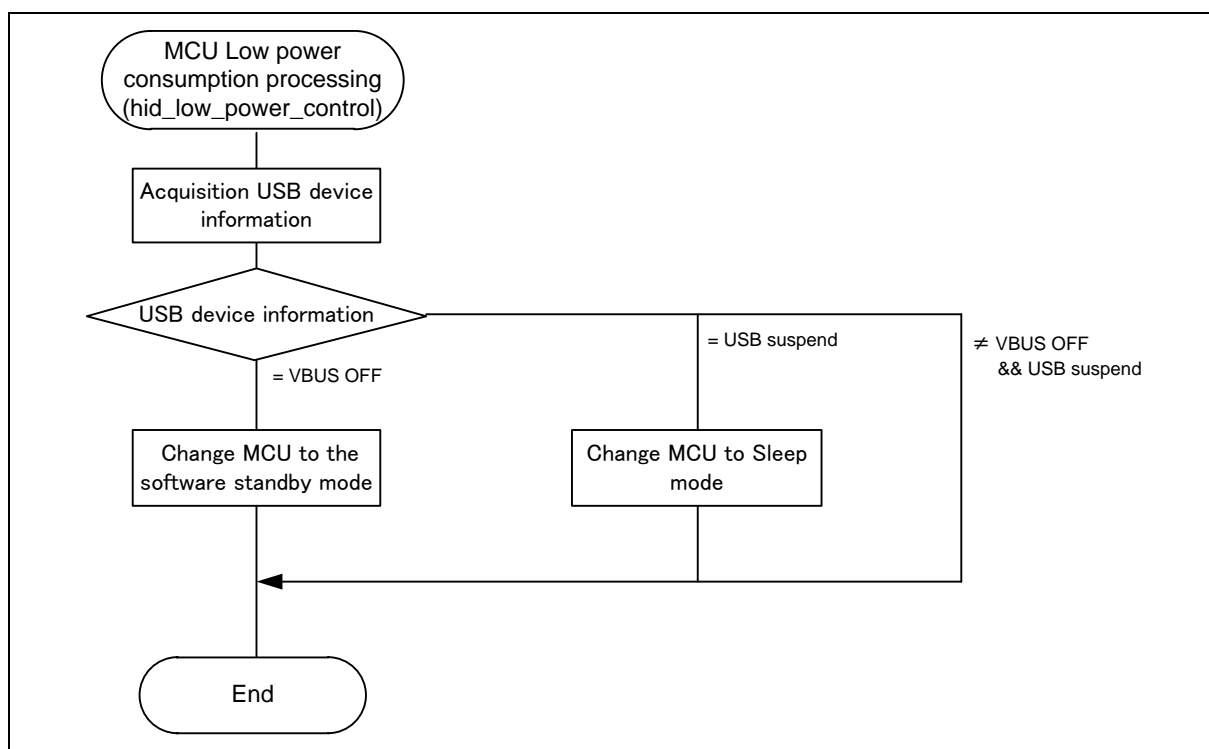



Figure 7-4 消費電力低減処理フロー

7.3 スイッチ

7.3.1 スイッチ仕様

マウスモードおよびキーボードモードで使用するスイッチについての仕様を以下に示します。なお、このアプリケーションプログラムでは、スイッチが押下されただけではそのスイッチの押下は認識されず、スイッチが押下→リリースされることにより、スイッチの押下が認識されます。

1) マウスモード

スイッチ番号	動作説明
スイッチ 1(SW1)	左クリック
スイッチ 2(SW2)	上下方向の移動データを通知します

スイッチ 3(SW3)	左右方向の移動データを通知します

2)　キーボードモード

スイッチ番号	動作説明
スイッチ 2(SW2)	スイッチを押すたびに“a”–“z”、“Enter”のキーコードを 1 つ通知します。
スイッチ 3(SW3)	スイッチを押すたびに“1”–“9”、“0”､“Enter”のキーコードを 1 つ通知します。

（注1）　スイッチを押し続けると APL が移動データ量を生成し、ホストへそのデータ量を通知します。スイッチが離されるとデータ通知を停止します。再度スイッチを押すと、移動方向を切り替え、移動データを通知します。

（注2）　スイッチ 2 およびスイッチ 3 が非押下の場合、キーボードモードでは USB ホストに対し NULL データが転送され、マウスモードでは USB ホストに対しデータ転送は行われません。

7.3.2　データフォーマット

USB ホストに転送するデータのフォーマットと USB から通知されるデータのフォーマットを下表に示します。これらのデータフォーマットは USB ホストに転送している HID のレポートディスクリプタの内容と併せて設定しています。

Table 7-4　ホストに通知するデータフォーマット

offset	マウスモード (3Bytes)	キーボードモード (8Bytes)
0	b0：Button 1 b1：Button 2 b2：Reserved	Modifier keys
1	X displacement	Reserved
2	Y displacement	Keycode 1
3		Keycode 2
4		Keycode 3
5		Keycode 4
6		Keycode 5
7		Keycode 6

Table 7-5　キーボード OUTPUT レポートフォーマット

offset (byte)	値
0	b0：LED 0 (NumLock) b1：LED 1 (CapsLock) b2：LED 2 (ScrollLock) b3：LED 3 (Compose) b4：LED 4 (Kana) 0: 消灯、1: 点灯

7.4　ディスクリプタ

PHID のディスクリプタ情報は r_usb_phid_descriptor.c に記述しています。

8. セットアップ

8.1 ハードウェア

PHID の動作環境例をFigure 8-1に示します。評価ボードのセットアップ、エミュレータなどの使用方法については各取扱説明書を参照してください。

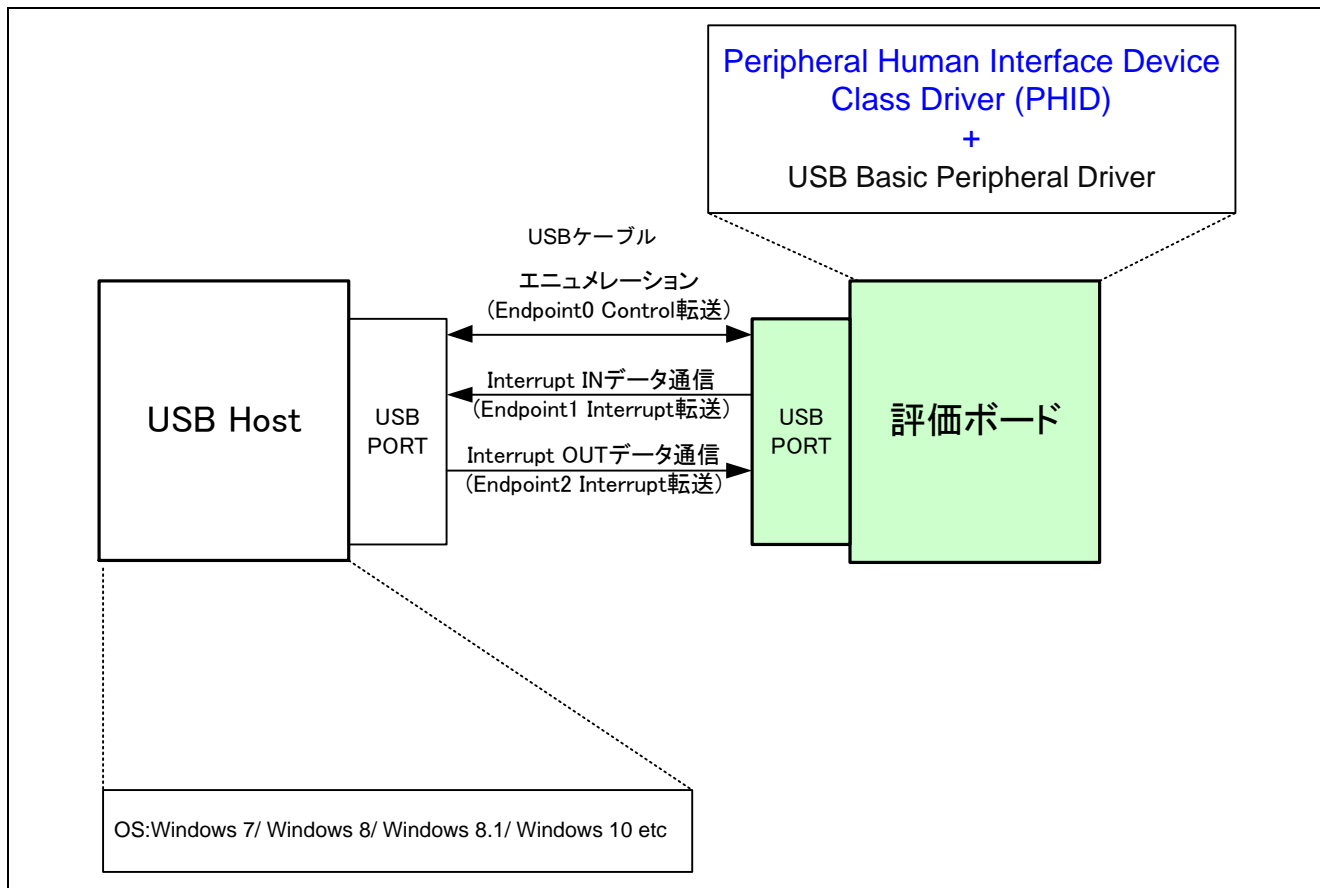
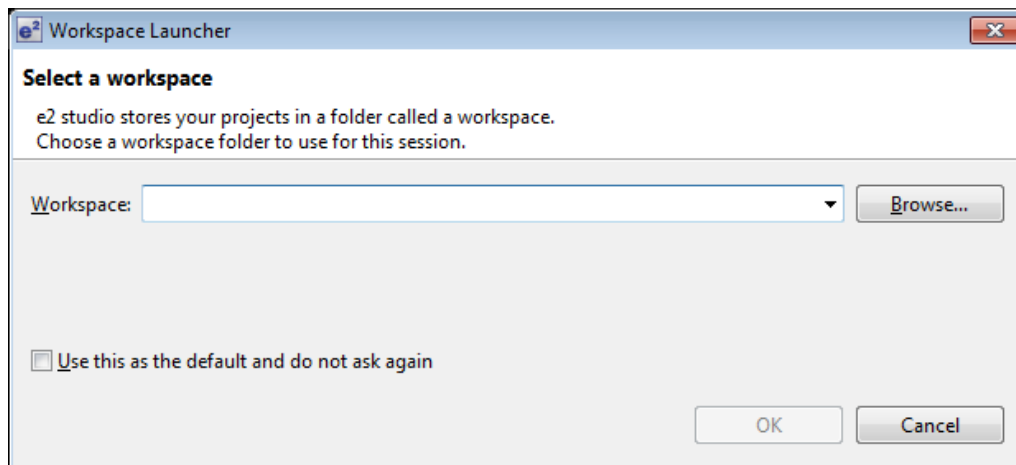



Figure 8-1 動作環境例

8.2 ソフトウェア

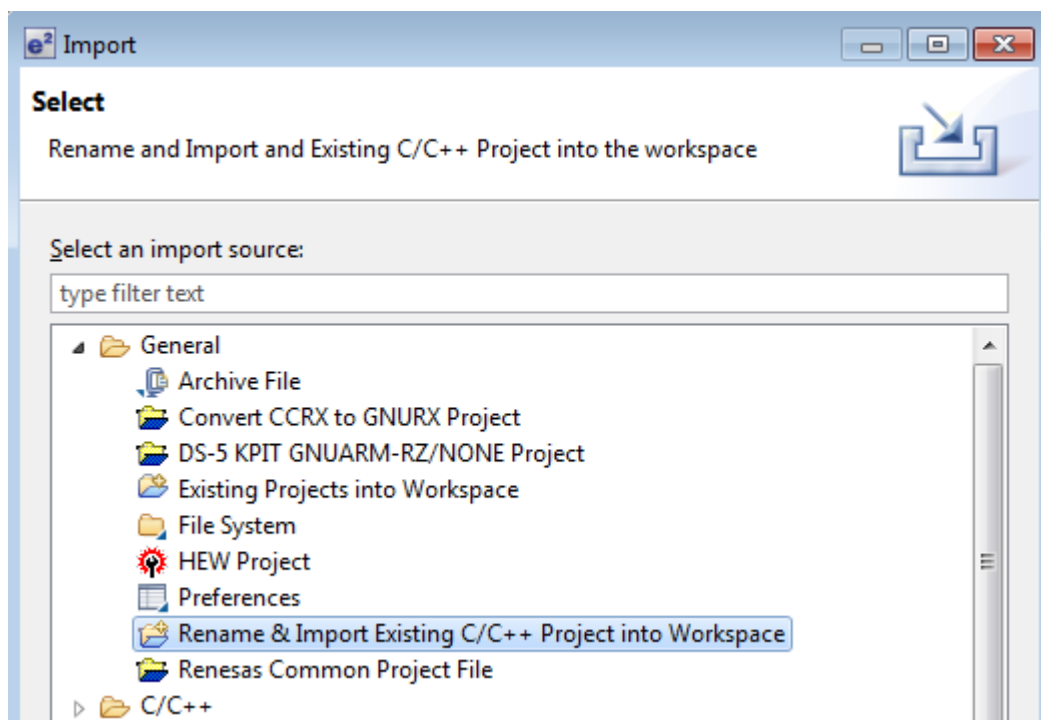
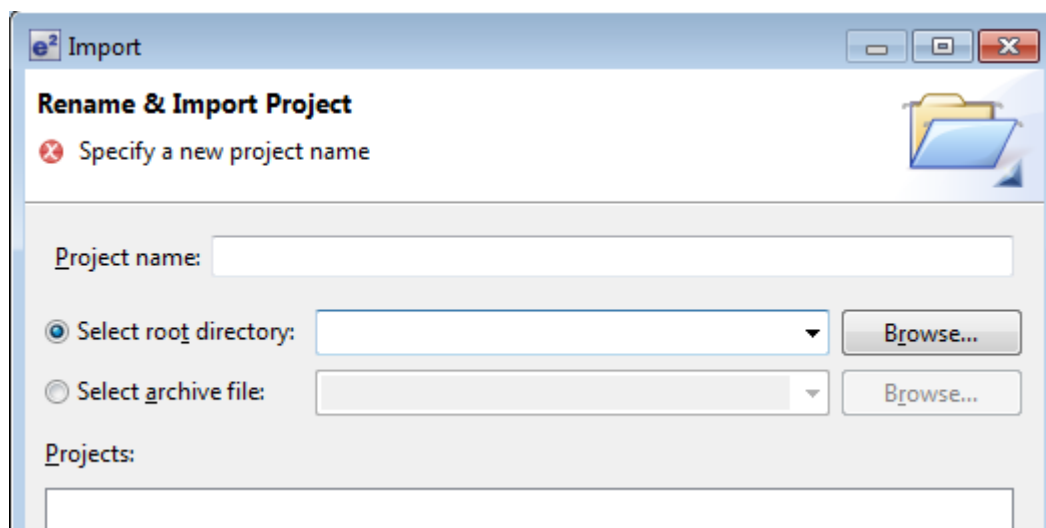
(1). e2 studio を起動

a) e2 studio を起動してください。

b) はじめて e2 studio を起動する場合、Workspace Launcher ダイアログが表示されますので、プロジェクトを格納するためのフォルダを指定してください。

c) Workbench アイコンをクリックしてください。

Welcome to e2studio

(2). ワークスペースへのプロジェクトの登録

a) [File] --> [Import]を選択してください。
b) General => Rename & Import Existing C/C++ Project into Workspace を選択してください。

プロジェクトファイル".cproject"が格納されたフォルダを"Select root directory"に入力してください。

c) "Finish"をクリック

プロジェクトのワークスペースへのインポートが完了しました。同様の方法で他のプロジェクトを同一のワークスペースへインポートすることができます。

(3). "Build"ボタンをクリックし、実行プログラムを生成してください。

(4). デバッガへの接続を行い、実行プログラムをダウンロードしてください。"Run"ボタンをクリックすると、プログラムが実行されます。

9. アプリケーションの作成方法

本章では、PHID とUSB-BASIC-FWを組み合わせ、USB ドライバとして使用するために必要な初期設定の方法と、メインルーチン処理方法及び API 関数を使用したデータ転送例を示します。

9.1 USB ドライバの初期設定方法

USB ドライバを使用するためには、以下の設定を行う必要があります。

- MCU の端子設定
- USB コントローラの起動と設定
- USB ドライバの設定

以下に各設定の例を示します。

```
/* PHID 用 USB 通信構造体 */
USB_UTR_t usb_gphid_utr

void usb_phid_apl(void)
{
        USB_ER_t        err;

        /* MCU の端子設定（「9.1.1 MCU の端子設定 」参照*/
        usb_mcu_setting();

        /* USB ドライバの設定（「9.1.2 USB ドライバの設定」参照） */
        ptr         = &usb_gphid_utr;
        ptr->ip   = USB_IPNUM;
        ptr->ipp = R_usb_cstd_GetUsbIpAdr(ptr->ip);

        usb_cstd_ScheInit();
        R_usb_pstd_PcdOpen();

        usb_phid_driver_registration();
        R_usb_phid_driver_start( ptr );

        /* USB モジュールの起動と設定（「9.1.3  USB モジュールの起動と設定」参照） */
        err = R_USB_Open();
        if(err != USB_SUCCESS)
        {
            /* エラー処理 */
        }
        R_usb_cstd_UsbIpInit(ptr, USB_PERI);

        /* メインルーチン */
        usb_papl_mainloop();
}
```

9.1.1 MCU の端子設定

USB コントローラを使用するためには、USB の入出力端子を設定する必要があります。以下に、設定が必要な USB 入出力端子例を示します。

Table9-1 ホスト動作時の USB 入出力端子設定

端子名	入出力	機能
USB_VBUS	入力	USB 用 VBUS 出力許可端子

※ PHID およびUSB-BASIC-FWは MCU の端子設定を行いません。端子設定の設定が必要な場合は各 MCU のユーザーズマニュアルを参照し、ご使用の評価ボードに合わせて端子設定を行ってください。

9.1.2 USB ドライバの設定

USB ドライバの設定では、スケジューラへのタスク登録及びUSB-BASIC-FWに対するクラスドライバの情報登録を行います。以下に、クラスドライバ情報の登録とスケジューラへのタスク登録手順を示します。

① USB_UTR_t 型で宣言された USB 通信構造体のメンバに USB モジュールの IP 番号を設定する。
② R_usb_cstd_GetUsbIpAdr()を呼び出し、USB 通信構造体に USB レジスタのベースアドレスを設定する。
③ USB-BASIC-FWの API 関数（R_usb_pstd_PcdOpen()）を呼び出し、スケジューラへ PCD タスクを登録する。
④ クラスドライバ登録用構造体（USB_PCDREG_t）の各メンバに情報を設定後、R_usb_pstd_DriverRegistration()を呼び出すことでUSB-BASIC-FWに対するクラスドライバの情報登録を行う。
⑤ クラスドライバの API 関数（R_usb_phid_driver_start()）を呼び出し、スケジューラへ PHID タスクを登録する。

USB_PCDREG_t で宣言された構造体への設定例を以下に示します。

```
void usb_phid_driver_registration( void )
{
    USB_PCDREG_t driver;    ←クラスドライバ登録用構造体

    /* パイプ情報テーブルを設定 */
    driver.pipetbl      = (uint16_t*)&usb_gphid_EpPtr; // (注 1)
    /* Device Descriptor テーブルを設定 */
    driver.devicetbl    = (uint8_t*)&usb_gphid_DeviceDescriptor;   // (注 2)
    /* Qualifier Descriptor テーブルを設定 */
    driver.qualitbl     = (uint8_t*)USB_NULL; // (注 3)
    /* Configuration Descriptor テーブルを設定 */
    driver.configtbl    = (uint8_t**)usb_gphid_ConPtr; // (注 2, 4)
    /* Other Configuration Descriptor テーブルを設定 */
    driver.othertbl     = (uint8_t**)USB_NULL; // (注 3)
    /* String Descriptor テーブルを設定 */
    driver.stringtbl    = (uint8_t**)usb_gphid_StrPtr; // (注 2, 5)
    /* クラスドライバ登録時に呼び出される関数を設定 */
    driver.classinit    = (USB_CB_INFO_t)&usb_cstd_DummyFunction;
    /* デフォルトステート遷移時に呼び出される関数を設定 */
    driver.devdefault  = (USB_CB_INFO_t)&usb_cstd_DummyFunction;
    /* エニュメレーション完了時に呼び出される関数を設定 */
    driver.devconfig   = (USB_CB_INFO_t)&usb_phid_smpl_open; // (注 6)
    /* USB デバイス切断時に呼ばれる関数を設定 */
    driver.devdetach  = (USB_CB_INFO_t)&usb_phid_smpl_close; // (注 6)
    /* デバイスをサスペンド状態に移行時に呼ばれる関数を設定 */
    driver.devsuspend           = (USB_CB_INFO_t)&usb_phid_suspend_cb;;
    /* デバイスのサスペンド状態解除時に呼ばれる関数を設定 */
    driver.devresume = (USB_CB_INFO_t)&usb_phid_smpl_resume_cb;
    /* インタフェース変更時に呼ばれる関数を設定 */
    driver.interface    = (USB_CB_INFO_t)& usb_cstd_DummyFunction;
    /* 標準リクエスト以外のコントロール転送処理時に呼ばれる関数を設定 */
    driver.ctrltrans    = (USB_CB_TRN_t)&R_usb_phid_control_transfer;

    /* PCD へクラスドライバ情報を登録 */
    R_usb_pstd_DriverRegistration(ptr, &driver);
}
```

（注1） パイプ情報テーブルは、アプリケーション内で定義してください。パイプ情報テーブル例を以下に示します。パイプ情報テーブルについては USB Basic Host and Peripheral アプリケーションノート (Document No. R01AN0512JJ)を参照してください。

= パイプ情報テーブル例 =

```
uint16_t usb_gphid_EpTbl[] =
{
    USB_PHID_USE_PIPE_IN,
    USB_INT | USB_DIR_P_IN | USB_EP1,
    USB_NONE,
    USB_PHID_MAXP,
    USB_NONE,
    USB_CUSE,

    USB_PHID_USE_PIPE_OUT,
    USB_INT | USB_DIR_P_OUT | USB_EP2,
    USB_NONE,
    USB_PHID_MAXP,
    USB_NONE,
    USB_CUSE,

    USB_PDTBLEND,
};
```

なお、このメンバにはパイプ情報テーブルを設定した配列の先頭アドレスを設定してください。下記参照。

```
uint8_t *usb_gphid_EpPtr[] =
{
    usb_gphid_EpTbl,
}
```

（注2） 各ディスクリプタはUSB規格書をもとに作成してください。
（注3） これらのメンバには”USB_NULL”を指定しください。
（注4） このメンバにはコンフィグレーションディスクリプタテーブルを設定した配列の先頭アドレスを設定してください。

```
uint8_t *usb_gphid_ConPtr[] =
{
    usb_gphid_Configration
};
```

（注5） このメンバにはストリングディスクリプタを設定した配列の先頭アドレスを設定してください。下記参照。

```
uint8_t *usb_gphid_StrPtr[] =
{
    usb_gphid_StringDescriptor0,
    usb_gphid_StringDescriptor1,
    usb_gphid_StringDescriptor2,
    usb_gphid_StringDescriptor3,
}
```

（注6） これらのメンバに登録した関数では、外部変数の初期化等を行ってください。

9.1.3 USB モジュールの起動と設定

以下に、USB モジュールの起動と設定手順を示します。

a. R_USB_Open()を呼び出すことでUSB モジュールを起動させます。本API 関数は初期設定時に一度だけ呼び出してください。
b. R_usb_cstd_UsbIpInit()を呼び出すことでUSB モジュールの初期設定を行います。

9.2 メインルーチン処理方法

USB ドライバは初期設定後アプリケーションのメインルーチン内でスケジューラ（R_usb_cstd_Scheduler()）を呼び出すことで動作します。

メインルーチン内で R_usb_cstd_Scheduler()を呼ぶことにより_usb_cstd_Scheduler()がイベントの有無を確認し、イベントがある場合、スケジューラにイベントが発生していることを通知するためのフラグをセットします。

R_usb_cstd_Scheduler()を呼び出し後、必ず R_usb_cstd_CheckSchedule()を呼び出しイベントの有無を確認してください。また、イベントの取得とそのイベントに対する処理は定期的に行う必要があります。（注 1）

```
void usb_papl_mainloop(void)
{
    while(1)        ←メインルーチン
    {
      R_usb_cstd_Scheduler();               ←イベントの確認と取得、フラグセット (注 1)    ┐
      if(USB_FLGSET == R_usb_cstd_CheckSchedule())    ←イベント有無の判定、フラグクリア  │
      {                                                                                   │ (注 2)
          R_usb_pstd_PcdTask();                  ←PCD タスク                              │
          R_usb_phid_task();                     ←HID ドライバタスク                      │
      }                                                                                   ┘
      phid_application();                   ←ユーザーアプリケーション
    }
}
```

（注1） R_usb_cstd_Scheduler()でイベントを取得後、処理を行う前に再度 R_usb_cstd_Scheduler()で他のイベントを取得すると、最初のイベントは破棄されます。イベント取得後は必ず各タスクを呼び出し、処理を行ってください。

（注2） これらの処理は、アプリケーションプログラムのメインループ内に必ず記述してください。

9.3 データ転送方法

アプリケーションから PHID のクラス API 関数（R_usb_phid_send_data () / R_usb_phid_receive_data()）を呼び出すことで、接続された USB デバイスに対してデータ送受信要求（Interrupt In/Out 転送）を行います。

R_usb_phid_send_data ()および R_usb_phid_receive_data ()は以下の引数を持ちます。

データ送信時

```
R_usb_phid_send_data (
        USB_UTR_t     *ptr          ←USB 通信構造体領域へのポインタ
        uint8_t       *Table,       ←送信データ格納領域のアドレス（注 1）
        uint32_t      size,         ←送信データサイズ
        USB_CB_t      complete      ←データ送信完了時に呼ばれるコールバック関数（注 1）
)
```

データ受信時

```
R_usb_phid_receive_data (
        USB_UTR_t     *ptr          ←USB 通信構造体領域へのポインタ
        uint8_t       *Table,       ←受信データ格納領域のアドレス（注 1）
        uint32_t      size,         ←受信要求サイズ
        USB_CB_t      complete      ←データ送信完了時に呼ばれるコールバック関数（注 1）
)
```

（注1） データ格納領域およびデータ転送完了時に呼ばれるコールバック関数はアプリケーションで定義する必要があります。

10. e² studio 用プロジェクトを CS+で使用する場合

PHID のプロジェクトは、統合開発環境 e² studio で作成されています。PHID を CS+で動作させる場合は、下記の手順にて読み込んでください。

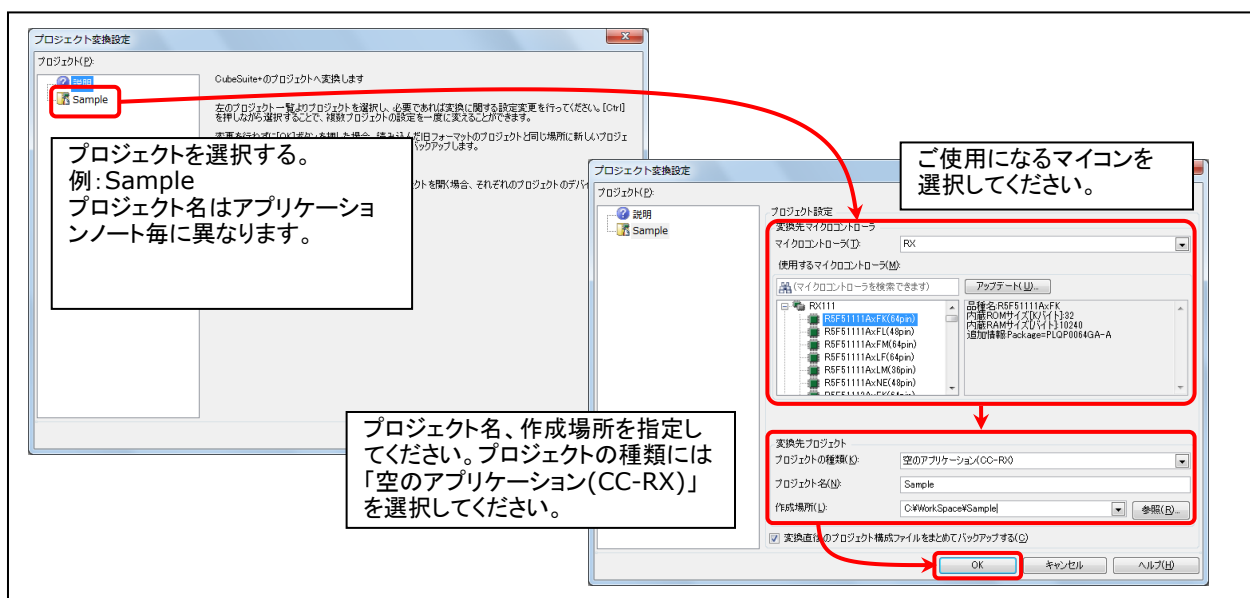
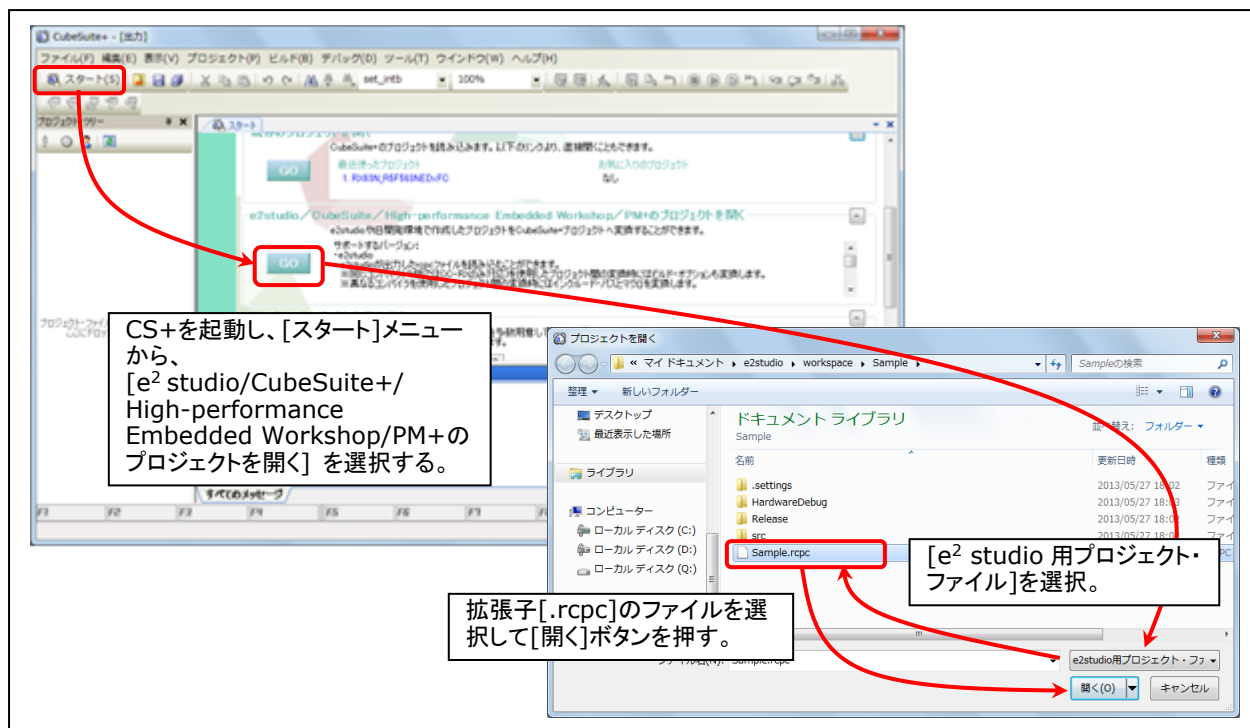



Figure 10-1 e² studio 用プロジェクトの CS+読み込み方法

ホームページとサポート窓口

ルネサス エレクトロニクスホームページ
http://japan.renesas.com/

お問合せ先
http://japan.renesas.com/inquiry

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2011.04.11	―	初版発行
1.10	2011.08.10	―	動作確認デバイス　RX630、R8A66597 の追加 上記追加に伴い、RX630 に関する内容、R8A66597(Hi-Speed)に関する内容を追加
		21	5.ペリフェラル用 HID サンプルアプリケーション(APL)のアプリケーション使用変更 ・SW1 使用禁止 ・SW 押下表記を機能スイッチ表記に変更
2.00	2012.09.30	―	ファームウエアアップデートによるドキュメントの改訂
2.01	2012.2.1	―	“1.5 本書の読み方”追加、"ファイル一覧"の変更
2.10	2012.4.1	―	V.2.10 用 First Release 動作確認デバイスに RX63T, R8A66593 を追加。この追加に伴いこれらのデバイスに関する内容を追加
2.20	2015.9.30	―	アプリケーションプログラムを変更 フォルダ構成を変更 対象デバイスに RX63N と RX631 を追加。 対象デバイスから R8A66597 と R8A66593 を削除

製品ご使用上の注意事項

ここでは、マイコン製品全体に適用する「使用上の注意事項」について説明します。個別の使用上の注意事項については、本文を参照してください。なお、本マニュアルの本文と異なる記載がある場合は、本文の記載が優先するものとします。

1. 未使用端子の処理

   【注意】未使用端子は、本文の「未使用端子の処理」に従って処理してください。

   CMOS製品の入力端子のインピーダンスは、一般に、ハイインピーダンスとなっています。未使用端子を開放状態で動作させると、誘導現象により、LSI周辺のノイズが印加され、LSI内部で貫通電流が流れたり、入力信号と認識されて誤動作を起こす恐れがあります。未使用端子は、本文「未使用端子の処理」で説明する指示に従い処理してください。

2. 電源投入時の処置

   【注意】電源投入時は，製品の状態は不定です。

   電源投入時には、LSIの内部回路の状態は不確定であり、レジスタの設定や各端子の状態は不定です。

   外部リセット端子でリセットする製品の場合、電源投入からリセットが有効になるまでの期間、端子の状態は保証できません。

   同様に、内蔵パワーオンリセット機能を使用してリセットする製品の場合、電源投入からリセットのかかる一定電圧に達するまでの期間、端子の状態は保証できません。

3. リザーブアドレスのアクセス禁止

   【注意】リザーブアドレスのアクセスを禁止します。

   アドレス領域には、将来の機能拡張用に割り付けられているリザーブアドレスがあります。これらのアドレスをアクセスしたときの動作については、保証できませんので、アクセスしないようにしてください。

4. クロックについて

   【注意】リセット時は、クロックが安定した後、リセットを解除してください。

   プログラム実行中のクロック切り替え時は、切り替え先クロックが安定した後に切り替えてください。

   リセット時、外部発振子（または外部発振回路）を用いたクロックで動作を開始するシステムでは、クロックが十分安定した後、リセットを解除してください。また、プログラムの途中で外部発振子（または外部発振回路）を用いたクロックに切り替える場合は、切り替え先のクロックが十分安定してから切り替えてください。

5. 製品間の相違について

   【注意】型名の異なる製品に変更する場合は、事前に問題ないことをご確認ください。

   同じグループのマイコンでも型名が違うと、内部メモリ、レイアウトパターンの相違などにより、特性が異なる場合があります。型名の異なる製品に変更する場合は、製品型名ごとにシステム評価試験を実施してください。

ご注意書き

1. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。お客様の機器・システムの設計において、回路、ソフトウェアおよびこれらに関連する情報を使用する場合には、お客様の責任において行ってください。これらの使用に起因して、お客様または第三者に生じた損害に関し、当社は、一切その責任を負いません。
2. 本資料に記載されている情報は、正確を期すため慎重に作成したものですが、誤りがないことを保証するものではありません。万一、本資料に記載されている情報の誤りに起因する損害がお客様に生じた場合においても、当社は、一切その責任を負いません。
3. 本資料に記載された製品データ、図、表、プログラム、アルゴリズム、応用回路例等の情報の使用に起因して発生した第三者の特許権、著作権その他の知的財産権に対する侵害に関し、当社は、何らの責任を負うものではありません。当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
4. 当社製品を改造、改変、複製等しないでください。かかる改造、改変、複製等により生じた損害に関し、当社は、一切その責任を負いません。
5. 当社は、当社製品の品質水準を「標準水準」および「高品質水準」に分類しており、
   各品質水準は、以下に示す用途に製品が使用されることを意図しております。
   標準水準：　コンピュータ、OA機器、通信機器、計測機器、AV機器、
   　　　　　　家電、工作機械、パーソナル機器、産業用ロボット等
   高品質水準：輸送機器（自動車、電車、船舶等）、交通用信号機器、
   　　　　　　防災・防犯装置、各種安全装置等
   当社製品は、直接生命・身体に危害を及ぼす可能性のある機器・システム（生命維持装置、人体に埋め込み使用するもの等）、もしくは多大な物的損害を発生させるおそれのある機器・システム（原子力制御システム、軍事機器等）に使用されることを意図しておらず、使用することはできません。 たとえ、意図しない用途に当社製品を使用したことによりお客様または第三者に損害が生じても、当社は一切その責任を負いません。 なお、ご不明点がある場合は、当社営業にお問い合わせください。
6. 当社製品をご使用の際は、当社が指定する最大定格、動作電源電圧範囲、放熱特性、実装条件その他の保証範囲内でご使用ください。当社保証範囲を超えて当社製品をご使用された場合の故障および事故につきましては、当社は、一切その責任を負いません。
7. 当社は、当社製品の品質および信頼性の向上に努めていますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は耐放射線設計については行っておりません。当社製品の故障または誤動作が生じた場合も、人身事故、火災事故、社会的損害等を生じさせないよう、お客様の責任において、冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、お客様の機器・システムとしての出荷保証を行ってください。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様の機器・システムとしての安全検証をお客様の責任で行ってください。
8. 当社製品の環境適合性等の詳細につきましては、製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制するRoHS指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。お客様がかかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
9. 本資料に記載されている当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器・システムに使用することはできません。また、当社製品および技術を大量破壊兵器の開発等の目的、軍事利用の目的その他軍事用途に使用しないでください。当社製品または技術を輸出する場合は、「外国為替及び外国貿易法」その他輸出関連法令を遵守し、かかる法令の定めるところにより必要な手続を行ってください。
10. お客様の転売等により、本ご注意書き記載の諸条件に抵触して当社製品が使用され、その使用から損害が生じた場合、当社は何らの責任も負わず、お客様にてご負担して頂きますのでご了承ください。
11. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを禁じます。

注1. 本資料において使用されている「当社」とは、ルネサス エレクトロニクス株式会社およびルネサス エレクトロニクス株式会社がその総株主の議決権の過半数を直接または間接に保有する会社をいいます。

注2. 本資料において使用されている「当社製品」とは、注１において定義された当社の開発、製造製品をいいます。

ルネサス エレクトロニクス株式会社

■営業お問合せ窓口

http://www.renesas.com

※営業お問合せ窓口の住所は変更になることがあります。最新情報につきましては、弊社ホームページをご覧ください。

ルネサス エレクトロニクス株式会社　〒135-0061　東京都江東区豊洲3-2-24（豊洲フォレシア）

■技術的なお問合せおよび資料のご請求は下記へどうぞ。
総合お問合せ窓口：http://japan.renesas.com/contact/